dBASE

จเร เลิศสุดวิชัย

คลิปเปอร์ 5.01
สร้างโพรซีเดอร์/ฟังก์ชัน (1)

ติดตามต่อกับคลิปเปอร์ โปรแกรมการสร้างฟังก์ชัน และโพรซีเดอร์

ตอนที่แล้วได้กล่าวถึงการประกาศใช้งานตัวแปรในคลิปเปอร์ ซึ่งในคลิปเปอร์ได้เริ่มมีรูปแบบโครงสร้างภาษาการใช้งานตัวแปรใกล้เคียงกับภาษาระดับสูงอื่น ๆ แล้ว ตัวแปรในคลิปเปอร์เริ่มมีขอบเขตการใช้งาน (scope) และช่วงชีวิต (life time) และในตอนนี้ก็จะเป็นเรื่องที่เกี่ยวข้องกับตัวแปรส่วนที่ยังเหลืออยู่ ซึ่งจะรวมทั้งการประกาศและสร้างฟังก์ชันและโพรซีเดอร์ขึ้นมาใช้งานเอง ซึ่งการส่งผ่านพารามิเตอร์ให้ฟังก์ชันและโพรซีเดอร์จะมีความสัมพันธ์กับวิธีการประกาศและการเรียกใช้งานด้วย

MEMVAR

คำสั่งนี้เป็นคำสั่งใหม่ในคลิปเปอร์ 5.0 ซึ่งมีไว้เพื่อแก้ไขปัญหาความสับสนในการอ้างชื่อตัวแปรกับชื่อฟิลด์ของดาต้าเบสที่มีชื่อเหมือนกัน โดยจะทำให้การอ้างชื่อดังกล่าวเป็นการอ้างถึงตัวแปรในหน่วยความจำแทน การใช้งานดังนี้

```
MEMVAR <memvar> [,<memvar>...]
```

คำสั่งนี้จะไม่สร้างตัวแปรขึ้นมา แต่จะทำหน้าที่เพียงแต่บังคับให้คลิปเปอร์มองการอ้างชื่อที่วางไว้หลังคำสั่ง MEMVAR เป็นการอ้างถึงตัวแปรในหน่วยความจำ ซึ่งคำสั่งนี้จะต้องวางไว้ก่อนการใช้คำสั่งการทำงานอื่น ๆ ของคลิปเปอร์ (เหมือนกับ LOCAL และ STATIC) ไม่เช่นนั้นจะเกิดข้อผิดพลาดในการแปลโปรแกรม ดังโปรแกรมรูปที่ 1

ตัวอย่างโปรแกรมที่ยกมามีการใช้คำสั่ง MEMVAR บอกคอมพายเลอร์ของคลิปเปอร์ว่าในตอนแปลถ้ามีการอ้างถึงชื่อ account เฉย ๆ ให้หมายถึงตัวแปร account ในหน่วยความจำ ในตัวอย่างมีการเปิดไฟล์ Clinets ซึ่งมีฟิลด์ account เหมือนกัน ในลูปมีคำสั่งให้พิมพ์ค่าในฟิลด์ account ของไฟล์ดาต้า Clinets และพิมพ์ค่าของ

```
MEMVAR account
account:= 1
USE Clients NEW
WHILE ! EOF ( )
      ? Clients -> account, account + +
      SKIP
ENDDO
```

รูปที่ 1

ตัวแปร account แล้วเพิ่มค่า account ขึ้นอีกหนึ่ง การประกาศ MEMVAR จะช่วยให้ไม่ต้องใช้การอ้างในลักษณะนี้ดังรูปที่ 2

ข้อดีของการใช้ MEMVAR นอกจากจะช่วยขจัดความกำกวมในการอ้างใช้ซึ่งยังช่วยกำจัดข้อความเตือนในระหว่างการคอมพายล์โปรแกรมด้วย /W ไว้ด้วย ซึ่งการประกาศนี้ใช้กับตัวแปร PRIVATE และ PUBLIC เท่านั้น

FIELD

คล้ายกับ MEMVAR แต่ตรงข้ามกัน โดยทำให้การอ้างชื่อหมายถึงฟิลด์ของดาต้าเบส และการประกาศก็เหมือนกับ MEMVAR, LOCAL และ STATIC การประกาศคำสั่ง FIELD ต้องอยู่เหนือคำสั่งการทำงานใด ๆ ของคลิปเปอร์ (ไม่ว่าจะเป็นภายในโมดูลไฟล์ .PRG ในฟังก์ชันหรือโพรซีเดอร์) FIELD มีการใช้งานดังนี้

```
FIELD <fieldname> [,<fieldname>...]
[IN <database>]
```

ตัวอย่างการใช้งานดังรูปที่ 3

ตัวอย่างในรูปที่ 3 นี้ก็จะคล้าย ๆ กับตัวอย่างในเรื่องของ MEMVAR ซึ่งหลังจากประกาศ FIELD แล้วชื่อเหล่านั้นจะหมายถึงฟิลด์ในดาต้าเบสไฟล์ นอกจากนั้นในการประกาศ FIELD ยังสามารถระบุได้ด้วยว่าชื่อฟิลด์ใดใช้กับดาต้าเบสไบต์ใด ตารางที่ 1 ต่อไปนี้เป็นคำสั่งที่มีผลเหมือนกันในการทำงาน แต่การประกาศด้วย FIELD จะทำให้การเขียนโปรแกรมสะดวกขึ้น

EXTERNAL

คำสั่งประกาศ EXTERNAL เป็นการกำหนดชื่อฟังก์ชันหรือโพรซีเดอร์

```
M -> account หรือ MEMVAR -> account อีก
```

รูปที่ 2

```
FIELD files, sizes, subdrive, per__used IN Notap
FIELD status, account IN Clients
CLS
USE Notap NEW
USE Clients NEW
WHILE (lients-) (!EOF( ))
        ? status, account
        IF Notap -> (EOR ( ))
        :    ? "        ", fields, sizes, LEFT (subdrive, 25), ;
        :    :             per__used
        :    :
        :    SKIP ALIAS Notap
        ENDIF
        SKIP
        IF EMPTY (ROW ( ) % 22)
          WAIT
          CLS
        ENDIF
ENDDO
```

รูปที่ 3

ตารางที่ 1

การประกาศใช้ FIELD	การใช้พอยน์เตอร์ Field	การใช้ชื่อ ALIS เป็นพอยน์เตอร์
FIELD files, sizes, ; subdrive per__used ; IN Notap	USE Notap NEW SELECT Notap	USE Notap NEW
files	Filed -> files	Notap -> files
sizes	Field -> sizes	Notap -> sizes
subdrive	Filed -> subdrive	Notap -> subdrive
per__used	Field -> per__used	Notap -> per__used

ในการอ้างอิงให้กับ Linker นำเอาออปเจ็กต์ไฟล์ของโมดูลที่มีชื่อฟังก์ชันหรือโพรซีเดอร์ตามหลังประกาศ EXTERNAL มาลิงค์รวมกับโปรแกรม รูปแบบการใช้งานดังนี้

```
EXTERNAL <Procname>
[, <Procname>...]
```

ในการเขียนโปรแกรมตามปกติมักจะมีการเรียกใช้งานฟังก์ชัน หรือโพรซีเดอร์ด้วยชื่อโดยตรง ซึ่งในขั้นตอนการลิงค์ออปเจ็กต์ไฟล์ของ Linker จะรวมทุกโมดูลที่มีการอ้างอิงดังกล่าวมารวมเป็นโปรแกรม แต่ในบางครั้งการเรียกใช้ฟังก์ชันหรือโพรซีเดอร์จะไม่ถูกเรียกโดยตรง ซึ่งชื่อของฟังก์ชันหรือโพรซีเดอร์จะยังไม่ทราบในขณะการแปลโปรแกรมต้นฉบับ (Compile Source Program) แต่จะทราบก็ต่อเมื่อโปรแกรมทำงานแล้ว ในลักษณะเช่นนี้การแปลโปรแกรมต้นฉบับก็จะไม่ทราบว่ามีการอ้างใช้ฟังก์ชันหรือโพรซีเดอร์เหล่านั้น ซึ่งจะมีผลต่อในขั้นตอนการลิงค์โปรแกรม จะไม่ดึงเอาออปเจ็กต์

โมดูลที่มีฟังก์ชันหรือโพรซีเดอร์มารวมด้วย ทำให้โปรแกรม .EXE ทำงานไม่ได้เพราะขาดเนื้อโปรแกรมนี้ ในส่วนที่ไม่ได้ลิงค์รวมเข้าไปแต่มีการอ้างเรียกใช้ ดังตัวอย่างโปรแกรมต่อไปนี้

```
FOR X:=1 TO 3
    ext:=ALTRIM (STR (X))
    DO Protext.
NEXT
หรือ
branch = "Listchoice ( )"
Op (& branch)
```

ในตัวอย่างแรกจะเห็นว่าในเนื้อโปรแกรมต้นฉบับจะไม่สามารถรู้ชื่อโปรแกรมที่จะถูกเรียกด้วยคำสั่ง DO ได้ แต่เมื่อโปรแกรมทำงานจึงจะทราบว่ามีโพรซีเดอร์ชื่อ Proc1, Proc2 และ Proc3 ถูกเรียกใช้งาน

เช่นเดียวกับตัวอย่างที่สอง จะเห็นว่าเมื่อโปรแกรมทำงานเท่านั้นจึงจะทราบว่ามีการเรียกใช้งานโพรซีเดอร์ Listchoice ( )

การแก้ปัญหานี้ทำโดยใช้คำสั่งประกาศ EXTERNAL ตามโปรแกรมต่อไปนี้

```
EXTERNAL PROC1, PROC2, PROC3
FOR X:=1 TO
    ext:=ALTRIM (STR (X))
    DO Proc & ext.
NEXT
หรือ
EXTERNAL LISTCHOICE
branch = "Listchoice ( )"
OP (& branch.)
```

สิ่งที่ต้องระวังโดยอาจต้องเอาชื่อฟังก์ชันหรือโพรซีเดอร์มาประกาศด้วย EXTERNATE เมื่อมีการใช้งานดังนี้

- โพรซีเดอร์หรือฟังก์ชัน (ทั้งของคลิปเปอร์หรือที่ผู้ใช้สร้างขึ้นมาเอง) ที่ถูกเรียกทางอ้อมผ่านทาง หรือตัวแปรที่ใช้มาโคร (Macro)

- โพรซีเดอร์หรือฟังก์ชันที่มีการเรียกใช้ในแบบฟอร์มของ REPORT และ LABEL ซึ่งไม่เคยมีการเรียกชื่อตรง ๆ ในโปรแกรมต้นฉบับ (Source Program)

- โพรซีเดอร์หรือฟังก์ชันที่มีการใช้งานในการสร้างดัชนีของดาต้าเบสไฟล์ ซึ่งไม่เคยมีการเรียกชื่อตรง ๆ ในโปรแกรมต้นฉบับ

- ACHOICE( ) DBEDIT( ) หรือ MEMOEDIT ซึ่งต้องใส่ชื่อฟังก์ชันหรือโพรซีเดอร์เข้าไปเป็นพารามิเตอร์

เราสามารถรวบรวมการประกาศ EXTERNAL เข้าด้วยกัน และเก็บเอาไว้ในไฟล์ที่เป็นเฮดเดอร์ (Header File) แล้วในโปรแกรมต้นฉบับใช้คำสั่ง #include ดึงเอาเฮดเดอร์ไฟล์เข้ามาแปลด้วยกัน ดังตัวอย่างไฟล์ชื่อ Externals.ch ดังนี้

```
//Externals.ch
EXTERNAL HARDER
EXTERNAL TONE
EXTERNAL MEMOTRAM
EXTERNAL STRTRAN
```

การประกาศ EXTERNAL ก็เช่นเดียวกับการประกาศ LOCAL, STATIC, FIELD และ MEMVAR คือ จะต้องประกาศอยู่เหนือคำสั่งการทำงานใด ๆ ของคลิปเปอร์และต้องอยู่ก่อน PARAMETER, PUBLIC และ PRIVATE แต่ในกลุ่มดังกล่าวสามารถสลับตำแหน่งได้

สรุปลำดับการประกาศใช้งานตัวแปรให้อีกครั้งดังนี้

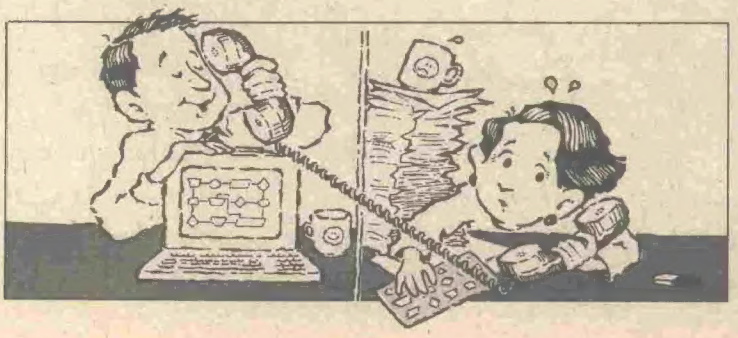
- FIELD, MEMVAR, EXTERNAL และ STATIC สามารถประกาศอยู่นอกฟังก์ชันและโพรซีเดอร์ได้แต่ต้องอยู่ส่วนบนของโมดูลคืออยู่ก่อนการประกาศฟังก์ชันและโพรซีเดอร์ใดๆ

- การประกาศภายในฟังก์ชันหรือโพรซีเดอร์มีลำดับการประกาศดังนี้

```
FIELD
MEMVAR
EXTERNAL
LOCAL
STATIC
PARAMETER
PRIVATE
PUBLIC
```

ดังตัวอย่างเช่น

```
Function MyFune
PARAMETER  Value
Value + +
LOCAL counter
count:=Value + +
RETURN (counter)
```



จากตัวอย่างนี้จะเห็นว่าการประกาศ LOCAL อยู่หลังคำสั่งการเพิ่มค่าให้กับตัวแปร Value ซึ่งไม่ถูกต้องและที่ไม่ถูกต้องยิ่งกว่านั้นก็คือ LOCAL จะต้องประกาศเอาไว้ก่อนการประกาศ PARAMETER ด้วย

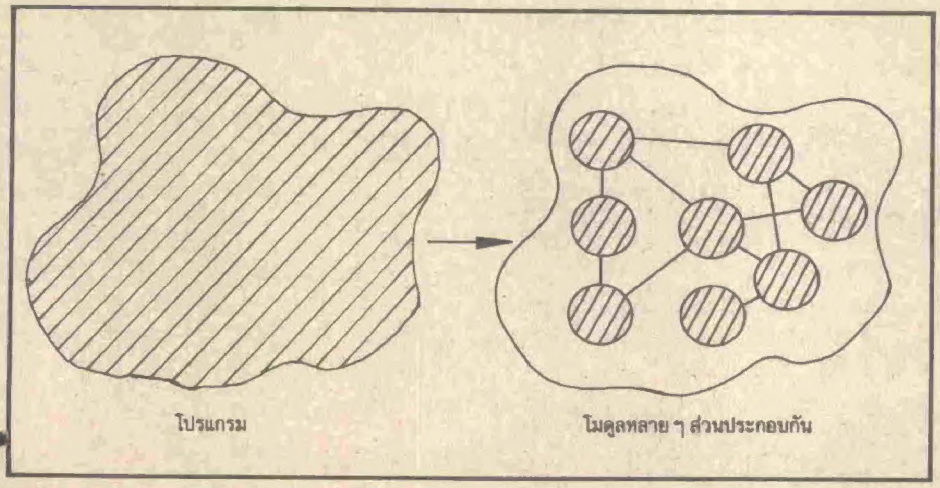
โพรซีเดอร์/ฟังก์ชันในคลิปเปอร์

ในคลิปเปอร์ 5.01 รูปแบบของภาษามีความใกล้เคียงกับภาษาโครงสร้างระดับสูงมากขึ้น ภาษาโครงสร้างการโปรแกรมคือการจัดเอากลุ่มคำสั่ง ซึ่งถูกตั้งชื่อเอาไว้มาประกอบกันเป็นโปรแกรม ในคลิปเปอร์ก็เช่นกันการสร้างโปรแกรมนั้นสร้างขึ้นมาจากการนำโพรซีเดอร์/ฟังก์ชัน (ที่ผู้ใช้สร้างขึ้นหรือที่มีอยู่แล้วในคลิปเปอร์) มาประกอบกันนั้นเอง ซึ่งต่อไปจะเป็นเรื่องเกี่ยวกับการประกาศสร้างโพรซีเดอร์/ฟังก์ชันรวมถึงการเรียกใช้งานโพรซีเดอร์และฟังก์ชันด้วย ในคลิปเปอร์เองพยายามที่จะคงหลายอย่างของคลิปเปอร์'87 เอาไว้เพื่อความคอมแพตทิเบิล ซึ่งมีผลทำให้เกิดทางเลือกหลายวิธีในการประกาศสร้างและใช้งานโพรซีเดอร์/ฟังก์ชัน สร้างความสับสนต่อผู้เขียนโปรแกรมที่อาจคุ้นเคยกับคลิปเปอร์'87 หรือกับผู้เขียนที่เริ่มต้นใหม่หลาย ๆ คน

แบ่งโปรแกรมออกเป็นส่วนๆ (หรือโมดูล)

ก่อนที่จะเข้าเรื่องของการประกาศสร้างและเรียกใช้งานโพรซีเดอร์/ฟังก์ชัน ขออนุญาตทบทวนวิธีการเขียนโปรแกรม ซึ่งจะมีหลายเรื่องในนี้เกี่ยวข้องกันอยู่ โดยตามปกติแล้วการสร้างโปรแกรม มักจะแบ่งโปรแกรมออกเป็นส่วน ๆ ที่ทำงานอิสระจากกัน ซึ่งเราเรียกส่วนของโปรแกรมเหล่านี้ว่า โมดูล โดยพยายามเก็บไว้ในไฟล์เดียวกัน ในกรณีของคลิปเปอร์ (ซึ่งก็เหมือนกับภาษาระดับสูงอื่น เช่น ภาษาซี ปาสคาล ฯลฯ) ในแต่ละโมดูลก็จะประกอบไปด้วย 2 ส่วนย่อย คือ ข้อมูลหรือตัวแปรที่ใช้สำหรับการทำงาน และส่วนของโพรซีเดอร์/ฟังก์ชัน เมื่อหลาย ๆ โมดูลมาประกอบกันเป็นโปรแกรมแต่ละโมดูลก็มักจะมีการเรียกใช้งานโพรซีเดอร์/ฟังก์ชันระหว่างโมดูลอื่น ๆ ซึ่งจะพิจารณาได้ดังรูปที่ 4

การเรียกใช้งานโพรซีเดอร์/ฟังก์ชันข้ามไปมาระหว่างโมดูลก็คือ การเชื่อมโยงของโมดูลนั้นเอง ซึ่งลักษณะการเชื่อมโยงก็จะต้องมีการกำหนดขอบเขตเพื่อปกป้องข้อมูล และป้องกันความสับสนของชื่อโพรซีเดอร์/ฟังก์ชัน ซึ่งอาจจะมีชื่อซ้ำกันในโมดูลอื่นได้ ลักษณะการปกป้องของโมดูลก็คือ การซ่อนบางส่วนของโมดูลเอาไว้ไม่ให้โมดูลอื่นเห็นหรือเข้าถึง หรือเรียกใช้ได้ การซ่อนก็จะทำกับข้อมูลหรือตัวแปรและโพรซีเดอร์/ฟังก์ชัน และถ้ายังจำได้การซ่อนลักษณะนี้ก็คือ การกำหนดขอบเขตการใช้งานนั้นเอง ในส่วนของตัวแปรเราสามารถกำหนดขอบเขตของตัวแปรให้อยู่ภายในโมดูลได้โดยประกาศคำสั่ง STATIC ในการประกาศชื่อตัวแปร และการกำหนดขอบเขตของโพรซีเดอร์/ฟังก์ชันก็สามารถทำได้ด้วยการประกาศคำสั่ง STATIC ไว้หน้าโพรซีเดอร์/ฟังก์ชัน



รูปที่ 4

การประกาศสร้างโพรซีเดอร์/ฟังก์ชัน

ในคลิปเปอร์ 5.0 นั้นการประกาศสร้างโพรซีเดอร์/ฟังก์ชัน ทำได้ 2 วิธี คือ ใช้รูปแบบใหม่ของคลิปเปอร์ 5.0 และรูปแบบเก่าของคลิปเปอร์'87 ซึ่งมีความแตกต่างกันในเรื่องพารามิเตอร์ที่ประกาศ ดังนี้

```
[STATIC] PROCEDERE <ชื่อโพรซีเดอร์> [(<โลคัลพารามิเตอร์>,...)]
:
: คำสั่งของคลิปเปอร์
[RETURN]
```

รูปที่ 5

```
[STATIC] FUNTION <ชื่อฟังก์ชัน> [(<โลคัลพารามิเตอร์>,...)]
:
: คำสั่งของคลิปเปอร์
RETURN <นิพจน์หรือค่าคงที่>
```

รูปที่ 6

```
PROCEDURE <ชื่อโพรซีเดอร์>
[<PARAMETER> <ชื่อพารามิเตอร์>,...]
:
: คำสั่งในคลิปเปอร์
[RETURN]
```

รูปที่ 7

```
FUNCTION <ชื่อฟังก์ชัน>
[<PARAMETER> <ชื่อพารามิเตอร์>,...]
:
:
RETURN <นิพจน์หรือค่าคงที่>
```

รูปที่ 8

รูปแบบใหม่

– การประกาศโพรซีเดอร์ ดังรูปที่ 5

– การประกาศฟังก์ชัน ดังรูปที่ 6

รูปแบบเก่า ('87)

– โพรซีเดอร์ ดังรูปที่ 7

– ฟังก์ชัน ดังรูปที่ 8

การประกาศทั้งสองแบบ ซึ่งสังเกตได้ชัดในเรื่องของการประกาศพารามิเตอร์มีความแตกต่าง และจะมีผลต่อพารามิเตอร์เหล่านั้น

การประกาศแบบใหม่ ซึ่งใช้การประกาศพารามิเตอร์ด้วยโลคัลพารามิเตอร์ พารามิเตอร์เหล่านั้นจะถูกสร้างขึ้นมาเสมือนเป็นตัวแปรแบบโลคัล คือ ขอบเขตหรือการอ้างใช้จะได้เพียงภายในโพรซีเดอร์/ฟังก์ชันเท่านั้น และช่วงชีวิตก็เท่ากับโพรซีเดอร์/ฟังก์ชัน ส่วนประกาศด้วยวิธีเก่าคือ ใช้คำประกาศ PARAMETER จะมีการสร้างพารามิเตอร์เป็นเหมือนตัวแปรพายเวต

การประกาศทั้งสองจะใช้ผสมกันภายในโพรซีเดอร์/ฟังก์ชันเดียวกันไม่ได้ (เช่น มีพารามิเตอร์บางตัวเป็นแบบโลคัลพารามิเตอร์บางตัวใช้ PARAMETER) เมื่อคอมพายล์โปรแกรมจะผิดพลาดโดยคลิปเปอร์ไม่คอยพายล์ต่อ

ในที่นี้ผู้เขียนขอเสนอเลยว่าควรใช้การประกาศโพรซีเดอร์/ฟังก์ชันแบบใหม่ในการเขียนโปรแกรม เพราะรูปแบบนี้เป็นสิ่งที่มีในภาษาคอมพิวเตอร์อื่นอยู่แล้ว และไม่ควรจะใช้ปนกันไป เช่น บางโพรซีเดอร์/ฟังก์ชันใช้แบบใหม่ บางโพรซีเดอร์/ฟังก์ชันก็ใช้แบบเก่า จะทำให้การตรวจสอบหรือติดตามการทำงานของโปรแกรมทำได้ยาก

ตัวอย่างการประกาศโพรซีเดอร์/ฟังก์ชันแบบเก่า ดังตัวอย่างนี้

```
FUNTION MyFune
PARAMETER cOne, cTWO, CThree
      ? cOne, cTwo, cThree
RETURN NIL
```

และสามารถประกาศเป็นแบบใหม่ได้ดังนี้ คือ

```
Function MyFunc (cOne, cTwo, CThree)
      ? cOne, cTwo, cThree
RETURN NIL
```

ในฉบับนี้ขอค้างเรื่องการเรียกโพรซีเดอร์/ฟังก์ชันเอาไว้ในฉบับหน้า ซึ่งการเรียกใช้งานก็เช่นเดียวกับการประกาศใช้ คือ คลิปเปอร์ 5.0 มีการเรียกใช้งานโพรซีเดอร์/ฟังก์ชันแบบใหม่ แต่ก็ยังยอมรับการเรียกใช้โปรแกรม'87 อยู่เช่นกัน และจะมีผลเกี่ยวข้องกับการส่งผ่านพารามิเตอร์ (passing parameter) ให้กับโพรซีเดอร์/ฟังก์ชันด้วย เอาไว้พบกันคราวหน้าครับ